🔷 Loops & Control Flow

1. Print all numbers between 1 and 500 divisible by 3 and sum of digits > 10

```
for i in range(1, 501):
    if i % 3 == 0:
        if sum(int(d) for d in str(i)) > 10:
            print(i, end=" ")
```

2. Find all Armstrong numbers between 1 and 10,000

```
for num in range(1, 10001):
    power = len(str(num))
    if num == sum(int(d)**power for d in str(num)):
        print(num, end=" ")
```

3. Print all prime numbers between 1 and N, skip numbers ending with 3

```
N = 100
for num in range(2, N+1):
    if str(num).endswith("3"):
        continue
    for i in range(2, int(num**0.5)+1):
        if num % i == 0:
```

```
        break
    else:
        print(num, end=" ")
```

4. Reverse a number without slicing using while loop

```
num = 12345
rev = 0
while num > 0:
    rev = rev*10 + num%10
    num //= 10
print(rev)
```

5. Print first N Fibonacci numbers using loops

```
N = 10
a, b = 0, 1
for _ in range(N):
    print(a, end=" ")
    a, b = b, a+b
```

🔷 Pattern Making

1. **Pyramid Pattern (N=5)**

```
N = 5
For i in range(1, N+1):
   Spaces = “ “ * (N-i)
   Num_list = list(range(i, 2*i))
   Num_list += list(range(2*i-2, i-1, -1))
   Print(spaces + “”.join(str(x) for x in num_list))
```

2. **Spiral Matrix NxN**

```
N = 3
Matrix = [[0]*N for _ in range(N)]
Top, left, right, bottom = 0, 0, N-1, N-1
Num = 1
While left <= right and top <= bottom:
   For i in range(left, right+1): matrix[top][i] = num; num += 1
   Top += 1
   For i in range(top, bottom+1): matrix[i][right] = num; num += 1
   Right -= 1
   For i in range(right, left-1, -1): matrix[bottom][i] = num; num += 1
   Bottom -= 1
   For i in range(bottom, top-1, -1): matrix[i][left] = num; num += 1
   Left += 1
For row in matrix: print(row)
```

3. **Diamond Pattern (*)**

```
N = 5
For i in range(1, N+1, 2):
   Print(“ “ * ((N-i)//2) + “*”*i)
For i in range(N-2, 0, -2):
   Print(“ “ * ((N-i)//2) + “*”*i)
```

4. **Pascal’s Triangle**

```
From math import comb
N = 5
For i in range(N):
   Print(“ “*(N-i), end=””)
   For j in range(i+1):
      Print(comb(i,j), end=” “)
   Print()
```

5. **Alphabet Triangle**

```
N = 4
For i in range(1, N+1):
   Print(“”.join(chr(65+j) for j in range(i)))
```

🔷 If-Logic & Match-Case

1. **Take marks and print Grade**

```
Marks = int(input(“Enter marks: “))

Match marks:
   Case m if m >= 90: print(“Grade A”)
   Case m if m >= 75: print(“Grade B”)
   Case m if m >= 60: print(“Grade C”)
   Case m if m >= 40: print(“Grade D”)
   Case _: print(“Grade F”)
```

2. **FizzBuzz (5 → Fizz, 7 → Buzz, both → FizzBuzz)**

```
For i in range(1, 51):
   If i % 5 == 0 and i % 7 == 0:
      Print(“FizzBuzz”)
   Elif i % 5 == 0:
      Print(“Fizz”)
   Elif i % 7 == 0:
      Print(“Buzz”)
   Else:
      Print(i)
```

3. **Calculator using match-case**

```
A = int(input(“Enter first number: “))
B = int(input(“Enter second number: “))
Op = input(“Enter operator (+,-,*,/): “)

Match op:
   Case “+”: print(a+b)
   Case “-“: print(a-b)
   Case “*”: print(a*b)
   Case “/”: print(a/b if b != 0 else “Error: Divide by zero”)
   Case _: print(“Invalid operator”)
```

4. **Leap year check**

```
Year = int(input(“Enter year: “))
If (year % 400 == 0) or (year % 4 == 0 and year % 100 != 0):
   Print(“Leap Year”)
Else:
   Print(“Not Leap Year”)
```

5. **Identify character type**

```
Ch = input(“Enter a character: “)

If ch.isalpha():
   If ch.lower() in “aeiou”:
      Print(“Vowel”)
```

```
    Else:
        Print("Consonant")
Elif ch.isdigit():
    Print("Digit")
Else:
    Print("Special Character")
```

🔷 Functions

1. GCD & LCM

```
Def gcd(a, b):
    While b:
        A, b = b, a % b
    Return a

Def lcm(a, b):
    Return a * b // gcd(a, b)

Print("GCD:", gcd(12, 18))
Print("LCM:", lcm(12, 18))
```

2. Second largest element (without sorted)

```
Nums = [10, 20, 4, 45, 99]
First = second = float('-inf')
For n in nums:
  If n > first:
    Second, first = first, n
  Elif n > second and n != first:
    Second = n
Print("Second largest:", second)
```

3. Palindrome string (ignore case & spaces)

```
S = "Never Odd Or Even"
Clean = s.replace(" ", "").lower()
Print("Palindrome" if clean == clean[::-1] else "Not Palindrome")
```

4. Multiplication table up to N

```
N = 5
For i in range(1, N+1):
  For j in range(1, 11):
    Print(f"{i} x {j} = {i*j}")
  Print()
```

5. Recursive sum of digits

```
Def sum_digits(n):
```

```
    If n == 0:
        Return 0
    Return n % 10 + sum_digits(n // 10)

Print(sum_digits(1234))  # 10
```

🔷 Functions

1. **GCD & LCM**

```
def gcd(a, b):
    while b:
        a, b = b, a % b
    return a

def lcm(a, b):
    return a * b // gcd(a, b)

print("GCD:", gcd(12, 18))
print("LCM:", lcm(12, 18))
```

2. **Second largest element (without sorted)**

```
nums = [10, 20, 4, 45, 99]
```

```
first = second = float('-inf')
for n in nums:
    if n > first:
        second, first = first, n
    elif n > second and n != first:
        second = n
print("Second largest:", second)
```

3. **Palindrome string (ignore case & spaces)**

```
s = "Never Odd Or Even"
clean = s.replace(" ", "").lower()
print("Palindrome" if clean == clean[::-1] else "Not Palindrome")
```

4. **Multiplication table up to N**

```
N = 5
for i in range(1, N+1):
    for j in range(1, 11):
        print(f"{i} x {j} = {i*j}")
    print()
```

5. Recursive sum of digits

```
def sum_digits(n):
    if n == 0:
```

```
    return 0
  return n % 10 + sum_digits(n // 10)

print(sum_digits(1234))  # 10
```

🔷 Strings

1. **All permutations of a string (without itertools)**

```
Def permute(s, ans=””):
  If len(s) == 0:
    Print(ans)
    Return
  For i in range(len(s)):
    Permute(s[:i] + s[i+1:], ans + s[i])

Permute(“ABC”)
```

2. **Longest palindrome substring**

```
Def longest_palindrome(s):
  Longest = “”
  For i in range(len(s)):
    For j in range(i, len(s)):
```

```
        Temp = s[i:j+1]
        If temp == temp[::-1] and len(temp) > len(longest):
          Longest = temp
    Return longest

Print(longest_palindrome(“babad”))
```

3. **Word frequency (ignore case)**

```
Text = “This is a test this is only a Test”
Words = text.lower().split()
Freq = {}
For w in words:
   Freq[w] = freq.get(w, 0) + 1
Print(freq)
```

4. **Anagram check (without Counter)**

```
S1, s2 = “listen”, “silent”
Print(“Anagram” if sorted(s1) == sorted(s2) else “Not Anagram”)
```

5. **Remove duplicate characters (preserve order)**

```
S = “programming”
Result = “”
For ch in s:
```

```
    If ch not in result:
        Result += ch
Print(result)
```

🔷 Lists

1. **Rotate list k times to the right without slicing**

```
Lst = [1, 2, 3, 4, 5]
K = 2

For i in range(k):
    Last = lst.pop()
    Lst.insert(0, last)

Print("Rotated list:", lst)
```

2. **Merge two sorted lists into one sorted list without sorted()**

```
A = [1, 3, 5, 7]
B = [2, 4, 6, 8]
Result = []

I = j = 0
```

```
While i < len(a) and j < len(b):
   If a[i] < b[j]:
      Result.append(a[i])
      I += 1
   Else:
      Result.append(b[j])
      J += 1

Result.extend(a[i:])
Result.extend(b[j:])

Print(“Merged sorted list:”, result)
```

3. **Find longest increasing subsequence in a list**

```
Lst = [10, 22, 9, 33, 21, 50, 41, 60]

N = len(lst)
Dp = [1] * n

For i in range(1, n):
   For j in range(0, i):
      If lst[i] > lst[j]:
         Dp[i] = max(dp[i], dp[j] + 1)
```

```
Print("Length of Longest Increasing Subsequence:", max(dp))
```

4. **Find all pairs in a list whose sum equals target**

```
Lst = [1, 2, 3, 4, 5, 6]
Target = 7

For i in range(len(lst)):
   For j in range(i+1, len(lst)):
      If lst[i] + lst[j] == target:
         Print("Pair:", (lst[i], lst[j]))
```

5. **Remove None and duplicates from a list**

```
Lst = [1, 2, None, 3, 2, None, 4, 1, 5]

Lst = [x for x in lst if x is not None]

Result = []
For x in lst:
   If x not in result:
      Result.append(x)

Print("Cleaned list:", result)
```

🔷 Tuples

1. Swap two tuples without extra variables

```
t1 = (1, 2, 3)
t2 = (4, 5, 6)

print("Before Swap:")
print("t1 =", t1)
print("t2 =", t2)

# swapping without extra variable
t1, t2 = t2, t1

print("After Swap:")
print("t1 =", t1)
print("t2 =", t2)
```

2**. Element-wise sum of two same-length tuples**

```
t1 = (1, 2, 3)
t2 = (4, 5, 6)

result = tuple(t1[i] + t2[i] for i in range(len(t1)))
```

```
print("Element-wise sum:", result)
```

3. Convert list of tuples into a dictionary

```
lst = [(1, "one"), (2, "two"), (3, "three")]

d = dict(lst)
print("Dictionary:", d)
```

4. **Count repeated elements in a tuple**

```
t = (1, 2, 3, 2, 4, 1, 2, 5)

counts = {}
for item in t:
    counts[item] = counts.get(item, 0) + 1

print("Element counts:", counts)
```

5. **Swap min and max elements in a tuple**

```
t = (10, 20, 5, 30, 15)
```

```
lst = list(t)   # convert tuple to list

min_index = lst.index(min(lst))
max_index = lst.index(max(lst))

# swap
lst[min_index], lst[max_index] = lst[max_index], lst[min_index]

t = tuple(lst)
print("After swapping min and max:", t)
```

🔷 Sets

1. **Find elements in exactly two out of three sets**

```
A = {1, 2, 3, 4}
B = {3, 4, 5, 6}
C = {4, 6, 7, 8}

# formula: (A∩B – C) ∪ (A∩C – B) ∪ (B∩C – A)
Result = (A & B – C) | (A & C – B) | (B & C – A)

Print(“Elements in exactly two sets:”, result)
```

2. **Check if two sets are disjoint without isdisjoint()**

A = {1, 2, 3}

B = {4, 5, 6}

Common = A & B # intersection

If len(common) == 0:

  Print("Sets are disjoint")

Else:

  Print("Sets are not disjoint")

3. **Find symmetric difference of two sets manually**

(Symmetric difference = elements in either set but not in both)

A = {1, 2, 3}

B = {3, 4, 5}

(A – B) ∪ (B – A)

Result = (A – B) | (B – A)

Print("Symmetric difference:", result)

4. **Create set of unique vowels in a string**

```
S = “Python Programming is Fun”

Vowels = “aeiouAEIOU”


Result = {ch for ch in s if ch in vowels}


Print(“Unique vowels:”, result)
```

5. **Generate set of prime factors of a number**

```
Num = int(input(“Enter a number: “))

Factors = set()


I = 2

Temp = num

While i <= temp:
   If temp % i == 0:
      Factors.add(i)
      Temp = temp // i
   Else:
      I = i + 1


Print(“Prime factors:”, factors)
```

🔷 Dictionary

1. **Character frequency**

```
S = “hello world”
Freq = {}
For ch in s:
   Freq[ch] = freq.get(ch, 0) + 1
Print(freq)
```

2. **Merge two dictionaries (sum common keys)**

```
D1 = {“a”:1, “b”:2}
D2 = {“b”:3, “c”:4}
Result = d1.copy()
For k,v in d2.items():
   Result[k] = result.get(k,0) + v
Print(result)
```

3. **Invert dictionary**

```
D = {“a”:1, “b”:2}
Print({v:k for k,v in d.items()})
```

4. **Group words by first letter**

```
Words = [“apple”,”banana”,”apricot”,”blueberry”]
Groups = {}
For w in words:
   Groups.setdefault(w[0], []).append(w)
Print(groups)
```

5. **Key with highest value**

```
D = {“a”:10, “b”:50, “c”:30}
Print(max(d, key=d.get))
```